Maximal Cliques 탐색 알고리즘들의 비교 및 분석
(Comparison and Analysis on the Maximal Clique Finding Algorithms)

이길행*조주현**
(G. H. Lee, J. H. Cho)

본고에서는 기존의 maximal cliques 탐색 알고리즘들을 조사하여 분석하고 문제점들을 제시하며 상호 비교분석함으로써 maximal cliques를 탐색하는 분야에 대한 알고리즘의 체계를 파악하고 기여할 수 있도록 노력하였다. 특히 기존의 clique 탐색 알고리즘들을 그들이 사용하는 기법에 따라서 point sequence method, line addition and removal technique, backtracking techique, 그리고 stack operation techique로 분류하고 각 기법에 해당하는 사례 알고리즘들을 분석하여 장단점들을 파악하며 상호 비교분석하는데 그 초점을 맞추었다.

Ⅰ. 서 론

Simple 그래프 G에 대해서 clique(또는 complete subgraph)란, S를 그래프 G의 vertex 집합의 부분집합이라고 할 때 그래프 G[S]가 complete 그래프인 것을 의미하며 S가 G의 clique가 되기 위한 필요충분 조건은 S가 G^c(G의 complement)의 independent이어야 한다는 것이다[1]. 따라서 maximal clique는 다른 어떤 complete subgraph에도 속하지 않는 clique를 의미하며, 주어진 그래프에서 maximal clique를 찾는 문제는 maximal independent set을 찾는 문제와 동일하게 된다. 이것은 주어진 그래프의 maximal clique가 보충(complementary) 그래프의 maximal independent set과 1대 1 대응관계에 있기 때문이고 clique를 찾는 문제는 independent set을 찾는 문제로 귀착될 수 있다.

Maximal clique를 찾는 문제는 다음과 같은 여러 분야에서 사용될 수 있다[4]. 첫째는 그래프 이론의 coloring 문제에 사용될 수 있다. 둘째, 교환 이론의 상태의 최소화에 사용될 수 있다. 상태의 최소화 문제는 각 요소가 최소한 한번은 발생하는 가장 작은 수의 부분집합을 찾는 것이며 이것은 곧 minimal cover를 찾는 문제이다. 셋째, scheduling을 다루는 OR(Operations Research)에서 최적의 처리결과를 만드는 데 사용된다. Scheduling 문제는 일정 기간 내에 모든 요소가 발생할 수 있는 가장

*운영체계연구실 선임연구원
**운영체계연구실 실장

작은 부분집합을 찾는 것이며 결국 minimal pair-cover를 찾는 문제가 된다. 그 밖에 node가 자료를 나타내고 edge가 자료간의 연관관계를 나타내는 정보 시스템에서도 사용될 수 있다[6].

많은 사람들이 maximal clique를 찾는 방법[2 ~ 8]들에 대해서 연구하였고, 이들중 Bierstone[2]이 처음으로 개발하였으며 Auguston과 Minker[3]의 알고리즘인 clustering 분석 기법에서 사용되었다. Auguston과 Minker는 Bierston의 원래 알고리즘을 약간 수정하였으며 그래프 이론적 clustering 기법들은 정보 조직에서 문제를 해결하는 데 사용되었다. 이런 clustering 기법들은 결국 각 vertex가 자료를 나타내고 각 edge가 두 자료간의 연결을 나타내는 그래프에서 clique를 찾는 것이 목적이다. Maximal clique를 찾는 방법들중에서 [3 ~ 6]은 상대적으로 큰 문제에 대해서 적용될 수 있다는 점에서 독특하게 여겨진다.

본 논문에서 사례로 분석하고 비교하는 clique 탐색 알고리즘들은 (그림 1)과 같이 분류될 수 있다. 첫째는 point sequence method로 그래프의 각 vertex를 point로 보고 각 point를 중심으로 clique를 찾는 방법이다. 여기에는 Bierston 알고리즘의 단점을 보완한 거의 유사한 알고리즘이다. 두번째는 line operation technique로 그래프 G에 해당하는 edge를 line으로 보고 G의 clique가 주어졌을 때 인접하는 두 vertex를 연결하여 clique를 탐색하는 line addition 방법, 그리고 인접하는 두 vertex를 연결하는 edge가 존재하는 그래프의 clique가 주어졌을 때 그 edge를 제거한 그래프 G의 clique를 탐색하는 line removal 방법이 있다. 이 방법에는 Robert의 알고리즘이 해당한다. 세번째는 backtracking technique로 탐색 clique에 이르지 못하는 branch를 사전에 배제함으로써 탐색과정을 최적화하기 위하여 branch-and-bound방법을 사용한다. 여기에는 Bron의 알고리즘이 해당한다. 마지막으로 stack operation 방법인데 그래프를 표현하는 수식을 스택을 이용하여 조직함으로써 크기가 큰 문제에 대해서도 효과적으로 대처할 수 있는 알고리즘이다. 이 방법에는 Akkoyunlu 알고리즘이 해당한다.

본 논문에서는 위에서 언급한 point sequence method, line operation technique, backtracking technique 그리고 stack operation technique에 해당하는 알고리즘들을 분석하여 그 특징들을 정리하고 상호 비교분석하였다. 본 논문의 구성은 다음과 같다. Ⅱ장에서는 위에서 언급한 maximal clique 탐색 알고리즘들의 사례들을 각 기법에 따라 분석하고, Ⅲ장에서는 clique 탐색 알고리즘의 분류와 각 알고리즘의 비교 및 분석을 다룬다. 마지막으로 Ⅳ장에서는 maximal clique 탐색 알고리즘의 분석에 대한 결론을 맺는다.

Ⅱ. Maximal Clique 탐색 알고리즘의 분석

1. Bierstone의 탐색 알고리즘

Bierstone의 알고리즘은 clique를 찾기 위한 최초의 방법으로 알고리즘의 입력은 upper triangular adjacency matrix M이다. 알고리즘은 matrix M을 한번에 한 열씩 거꾸로 인접한 vertex를 찾으면서 탐색한다.

$$m_{ij}\begin{cases} =1 & \text{vertex i가 j에 인접한 경우, } j>i \\ =0 & \text{그밖의 경우.} \end{cases}$$

알고리즘에서 사용하는 변수들에 대한 정의는 다음과 같다.

P_k : k번째 vertex를 의미한다.

M_j : j보다 큰 k에 대해서 vertex P_j에 연결된 모든 vertex P_k

C_i : 탐색된 i번째 complete subgraph

W : M_j중 처리되었으나 C_i에 포함되지 않은 vertex를 저장

S, T : 임시 기억장소

□ 알고리즘

1 단계 : i=0, j=입력되는 vertex수로 초기화.

2 단계 : j=j−1

3 단계 : M_j=0 이면 2단계 수행.

4 단계 : M_j에 속하는 각 P_k에 대해 i=i+1, $C_i=\{P_j, P_k\}$

5 단계 : j=j−1

6 단계 : j=0이면 C는 maximal clique를 의미함, j≠0이면 7단계 수행.

7 단계 : M_j=0이면 5단계 수행,
그렇지 않으면 W=M_j, L=i, n=0, 8단계 수행.

8 단계 : n=n+1

9 단계 : n>L이면 clique C_k이 탐색되었고 17 단계 수행,
그렇지 않으면 10 단계 수행.

10 단계 : $T=C_n \cap M_j$, T가 2 vertex 이하이면 8 단계 수행,
그렇지 않으면 W에서 T와 W에 모두 존재하는 vertex를 제거하고 11 단계 수행.

11 단계 : T=C_n이면 15 단계 수행.

12 단계 : T=M_j이면 i=i+1, $C_i=T \cup \{P_1\}$5 단계 수행,
그렇지 않으면 13 단계 수행.

13 단계 : T가 C_q의 부분집합이면 무시하고 8 단계 수행,
그렇지 않으면 $S=T \cup \{P_j\}$로 하고 14 단계 수행.

14 단계 : C_q가 S의 부분집합이면 C_q=S, S의 부분집합인 C_r 제거,
그렇지 않으면 i=i+1, C_i=S, 8 단계 수행.

15 단계 : $C_n=C_n \cup \{P_j\}$, C_n의 부분집합인 C_q제거.

16 단계 : T=M_j이면 5 단계 수행,
그렇지 않으면 8 단계 수행.

17 단계 : W에 남아있는 각 P_k에 대해서 i=i+1, $C_i=\{P_k, P_k\}$로 정의하고 5 단계 수행.

□ Bierstone 알고리즘의 분석

위에서 언급한 Bierstone의 알고리즘은 Bonner의 Cluster-Building 알고리즘보다 처리시간 측면에서 훨씬 좋은 향상을 보였다[3]. 그러나 Bierstone의 알고리즘은 maximal이 아닌 clique가 생성되거나 바로 이전에 생성된 subgraph의 부집합인 clique를 누락시키지만 그대로 무시한다. 또한 고립된 point들에 대해서는 clique를 생성하지 못한다. Bierstone의 알고리즘은 비교적 큰 clique들이 존재하는 그래프의 적용에 알맞다.

2. Gordon의 탐색 알고리즘

Gordon의 알고리즘은 Auguston과 Minker의 알

고리즘이 포함하는 에러를 수정한 것으로 adjacency matrix M이 0인 경우와 maximal clique가 아닌 clique를 제거하는 부분이 추가되었다. 따라서 Bierstone의 알고리즘과 다른 내용은 다음과 같다.

□ 알고리즘

2 단계 : j=j−1, j=0이면 M=0이므로 maximal clique가 없는 것으로 생각하고 종료.

12 단계 : $T=M_j$ 이면 $i=i+1, C_i=T\cup\{P_j\}$
C_i의 부분집합인 C_q을 제거하고 5 단계를 수행,
$T\neq M_j$이면 13 단계를 수행.

□ Gordon알고리즘의 분석

Gordon의 알고리즘은 Bierstone 알고리즘의 2 단계에 그래프의 vertex수를 조사하는 과정을 삽입함으로써 알고리즘이 끝나지 않는 위험을 배제하였고, 12 단계에 clique가 maximal일 경우에 이미 생성된 clique중에서 부분집합이 되는 clique를 제거하는 과정을 추가함으로써 maximal이 아닌 clique가 생성되는 오류를 처리하였다. Gordon의 알고리즘은 비교적 큰 clique를 갖는 그래프에 잘 적용된다. 그러나 Gordon의 알고리즘은 고립된 point들은 clique로 생성하지 못한다.

3. Bron의 탐색 알고리즘

Bron은 clique를 찾는 방법으로서 두 가지의 backtracking 알고리즘을 사용하였으며, 처리시간을 단축하기 위해서 clique에 이르지 못하는 branch를 미리 제거하는 branch-and-bound 기법을 사용하였다. 첫번째 방법은 기본 알고리즘의 간단한 구현으로 사용되는 방법들을 설명하기 위한 것이며 알파벳 순으로 clique를 생성한다. 두번째 방법은 첫번째 방법으로부터 파생되고 탐색되는 branch의 수를 최소화 할 수 있도록 예측하지 못하는 임의 순서로 clique들을 생성한다. 또한 첫번째 방법에 비해서 큰 clique를 생성하고 보다 큰 공통 부분을 갖는 clique들을 순차적으로 생성하는 경향이 있다.

Bron의 알고리즘에 사용되는 세가지의 set의 의미는 다음과 같다.

- compsub : 새로운 point에 의해서 확장되거나 backtracking 과정에서 임의의 point에 의해 줄어드는 set compsub를 확장하기 위해서 선택될 수 있는 point(compsub에서 모든 point들과 연결된 point)들은 나머지 두 set에서 반복적으로 모아진다.
- candidate : 현재의 compsub를 확장시키는 데 사용될 수 있는 모든 point의 set.
- not : 이전 단계에서 compsub의 확장에 사용되고 현재는 제외되어 있는 모든 point의 set.

□ 알고리즘

알고리즘의 핵심은 세 가지의 set에 적용될 반복적으로 정의되는 확장 연산자로 구성된다. 확장 연산자는 candidate의 set을 사용하여 not에 있는 어떤 point도 포함되지 않도록 현재의 compsub를 확장한다. 알고리즘의 동작은 다음과 같다.

1 단계 : candidate 선택

2 단계 : candidate를 compsub에 추가

3 단계 : 이전의 set에서 선택된 candidate에 연결

되지 않은 모든 point들을 제거함으로써 새로운 candidate와 not을 생성

4 단계 : 생성된 set 상에서 연산이 이루어지도록 확장 연산자를 호출

5 단계 : return하면서 선택된 candidate들을 compsub에서 제거하고 이전의 not set에 추가.

▫ Bron 알고리즘의 분석

알고리즘에서 candidate가 더 이상 없으면 compsub가 확장될 수 없으므로 새로운 clique가 생성될 필요조건이 됨을 알 수 있으며, compsub는 not과 candidate가 공집합이 되자마자 clique가 됨을 알 수 있다. 또한 not이 candidate에 있는 모든 point들과 연결된 point를 포함한다면 not으로부터 결코 제거되지 않으며 clique에 도달하지 못하게 하는 branch-and-bound 방법때문에 초기 단계에서 찾을 수 있다. 그리고 모든 clique는 알파벳 순으로 candidate의 초기 순서에 따라 생성된다.

알고리즘이 생성하는 clique의 비율은 그래프의 크기에 관계가 없으며 3*K의 regular 그래프에서 Bierstone의 알고리즘보다 성능이 훨씬 더 좋음을 나타냈다. 또한 고립된 point들도 clique로 생성된다.

4. Akkoyunlu의 탐색 알고리즘

Akkoyunlu는 maximal clique를 찾고 문제의 크기에 의해서 발생하는 문제점들을 다루었다. 대다수의 maximal clique를 찾는 알고리즘들이 큰 문제를 효과적으로 처리하지 못하는 원인은 중복되어 생성되는 clique나 submaximal 집합 때문이다. 이런 문제를 회피하려면 모든 list를 유지하고 반복적으로 검색하여야 한다.

▫ 그래프의 표현

Akkoyunlu는 알고리즘에서 사용할 자료구조로 그래프를 vertex를 포함하는 테이블의 구조로 표현하고 인접하는 vertex 사이에는 x로 표시함으로써 나타내는 방법과 각 vertex x에 대해서 인접한 vertex set C_x와 인접하지 않는 vertex set D_x를 동시에 테이블 형태로 표현하는 방법을 사용하였다. S가 S_0(vertex set)의 부분집합일 때 L(S)와 E(S)를 다음과 같이 정의한다.

L(S) : S의 요소를 적어도 1개 포함하는 모든 maximal subgraph의 set

E(S) : S의 모든 요소를 포함하는 모든 maximal subgraph의 set.

▫ 알고리즘

1 단계 : $L(S_0)$를 stack에 넣는다.

2 단계 : (a) stack이 비어 있으면 종료.

(b) stack에서 가져와 T에 넣는다.

3 단계 : S_k가 가장 작은 요소를 갖는 k를 선택.

S_k에 속하는 x에 대해 $S=S_k-\{x\}$

S가 비었으면 5 단계 수행.

4 단계 : $Q=D_x$, V가 비어있는 경우,

$Q=D_x \cap (\bigcap_{y \in V} C_y)$, 그렇지 않은 경우.

Q가 비어있는 경우는 5 단계 수행,

$T=L(S) \cap L(Q)$, $L(S_k)$를 대체하고 stack에 넣음.

5 단계 : $J=\{j \mid j \epsilon I, x \epsilon S_j\}$, J가 비어있으면 6 단계 수행.

J에 속하는 모든 j에 대해 $W_j=S_j\cap C_x$.

W_j이 비어있으면 2 단계 수행.

$E(V\cap\{x\})\cap(\bigcap_{j\in J}(L(W_j))$를 stack에 넣고

2 단계 수행.

6 단계 : $P=\{\bigcap_{y\in V\cup\{x\}}\}C_y$

P=0가 V{x}를 출력하려면 2 단계 수행

$E(V\cup\{x\})\cap L(P)$를 stack에서 가져오고

2 단계 수행.

□ Akkoyunlu 알고리즘의 분석

Akkoyunlu 알고리즘은 생성되는 set들의 중복이 없으면 이전에 생성된 set들을 참조하거나 유지할 필요가 없다. 이것은 maximal subgraph set이 겹치지 않도록 작은 문제로 체계적으로 줄이기 때문이다. 또한 크기가 큰 문제에 대해서도 기억공간의 크기에 큰 영향을 받지 않는다. Akkoynulu알고리즘의 특징은 stack으로 구성되어 있으며 핵심 요구사항이 최소화되어 있는데 있다.

□ Robert의 탐색 알고리즘

Robert는 그래프의 선과 점이 주어지는 adjacency 정보로부터 clique들을 찾아내는 두개의 알고리즘을 제시하였다. 첫번째 알고리즘은 같은 점들을 갖는 subgraph의 clique들과 subgraph에 없는 그래프의 선의 집합들을 사용한다 – line addition 알고리즘.

두번째 알고리즘은 같은 점들을 갖는 그래프의 subgraph의 clique들과 그래프에 없는 subgraph의 선의 집합들을 사용한다 – line removal 알고리즘.

□ line addition 알고리즘

그래프 G=(V,E)가 인접하지 않는 두점 x와 y를 포함한다면 두점 x와 y를 연결하는 선을 갖는 그래프 $G^*=(V,E^0)$, $E^0=E\cup\{xy\}$를 정의한다. Line addition 알고리즘은 G의 clique가 주어졌을 때 G^0의 clique를 찾는 방법이다.

□ line removal 알고리즘

Line removal 알고리즘은 인접하지 않는 두점 x와 y를 포함하는 그래프 G가 G=(V, E)로 정의되고 두점 x와 y를 연결하는 선을 포함하는 그래프 G^0가 $G^0=(V, E^0)$, $E^0=E\cup\{xy\}$로 주어지고 G^0의 clique가 주어졌을 때 G의 clique를 찾는 방법이다.

□ Robert 알고리즘의 분석

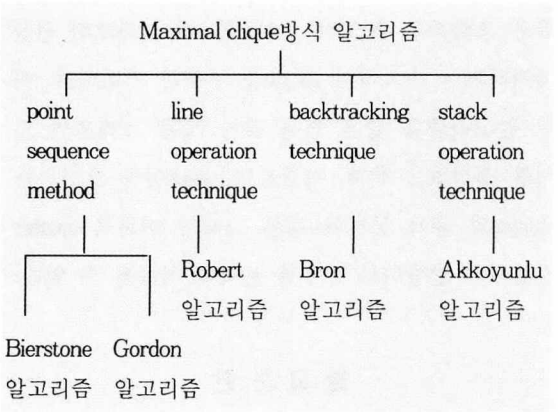
Robert은 알고리즘은 그래프의 clique가 주어졌을 때 그래프의 선을 제거하거나 추가하여 clique를 구하는 방법이다. 이런 알고리즘은 같은 점을 갖는 모든 그래프의 순서를 생성하는 문제, 그래프의 clique들을 요구하는 cluster의 분석 문제와 같은 그래프 이론적 cluster방법에서 발생하는 특수한 문제에 적용될 수 있다.

Ⅲ. Maximal clique 알고리즘들의 비교 분석

1. 알고리즘들의 분류

지금까지 본 눈문에서 사례연구로써 분석한 maximal clique를 찾는 알고리즘들은 그들이 사용한 방법에 따라서 (그림 1)과 같이 분류될 수 있다. 첫째는 point sequence method로 그래프와 각

vertex를 point로 보고 각 point를 중심으로 순차적으로 clique를 찾는 방법이다. 여기에는 Bierstone과 Gordon의 알고리즘이 해당되며, Gordon 알고리즘은 Bierstone 알고리즘의 단점을 보완한 거의 유사한 알고리즘이다. 두번째는 line operation technique로 그래프 G에 해당하는 edge를 line으로 보고 G의 clique가 주어졌을 때 인접하지 않는 두 vertex를 연결하여 clique를 탐색하는 line addition 방법, 인접하는 두 vertex를 연결하는 edge가 존재하는 그래프의 clique가 주어졌을 때 그 edge를 제거한 그래프 G의 clique를 탐색하는 line removal 방법이 있다. 이 방법에는 Robert의 알고리즘이 해당한다. 세번째는 backtracking technique로 탐색 clique에 이르지 못하는 branch를 사전에 배제함으로써 탐색 과정을 최적화하기 위하여 branch-and-bound 방법을 사용한다. 이 방법에는 Bron의 알고리즘이 해당한다. 마지막으로 stack operation 방법인데 그래프를 표현하는 수식을 스택을 이용하여 조정함으로써 크기가 큰 문제에 대해서도 효과적으로 대체할 수 있는 알고리즘이다. 이 방법에는 Akkoyunlu 알고리즘이 해당한다.



(그림 1) Maximal clique방식 알고리즘의 분류

지금까지 maximal clique 알고리즘의 사례연구로써 분석하고 비교한 내용들을 각 알고리즘이 적용한 그래프, 사용한 기법, 공집합 그래프의 처리능력, 크기가 큰 문제에 대한 처리능력, 고립된 vertex의 처리능력과 알고리즘의 복잡도 분석으로써 처리시간과 기억공간의 사용량 등의 관점에서 정리하면 〈표 1〉과 같다. 〈표 1〉에서 V는 탐색되는 clique의 수를 의미하며 M은 가장 큰 connect component의 크기를 의미한다.

〈표 1〉 Maximal clique 탐색 알고리즘의 비교표

사용한 기법	point sequence technique	backtracking tech.	stack operation tech.	line operation tech.
적용 가능한 그래프	undir. graph	undir. graph	undir. graph	undir. graph
공집합 그래프 처리능력	없음	처리가능	처리가능	처리가능
큰문제에의 적용성	약 1000 vertex처리	처리가능	처리가능	처리가능
고립된 vertex처리	처리 불가능	처리가능	처리가능	처리가능
처리 시간	V	?	?	V~V/8
기억공간	clique수에 따라 달라짐	0.5M(M+3)	알고리즘에 큰 영향을 미치지 못함	Akkoyunlu와 유사

위의 〈표 1〉에서 살펴본 바와 같이 본 논문에서 분석한 모든 알고리즘이 undirected그래프에 적용할 수 있는 내용이며 처리시간은 대체적으로 그래프에서 탐색되는 clique의 수에 의존함을 알 수 있다. 그리고 최초의 maximal clique탐색 알고리즘인 Bierstone 알고리즘을 제외하고는 모든 공집합을 갖는 그래프나 고립된 vertex의 처리가 가능함을 알 수 있다.

Ⅳ. 결 론

지금까지 본 논문에서는 maximal clique를 찾는 알고리즘의 사례연구로써 point sequence, backtracking, stack operation 그리고 line operation technique로 분류된 여러가지의 알고리즘들을 소개하고 분석하였다. Point sequence technique는 그래프의 각 vertex를 point로 보고 각 point를 중심으로 순차적으로 clique를 찾는 방법이다. 여기에는 Bierstone과 Gordon의 알고리즘이 해당하며 Gordon 알고리즘은 Bierstone 알고리즘의 단점을 보완한 거의 유사한 알고리즘이다. Line operation technique는 그래프 G에 해당하는 edge를 line으로 보고 G의 clique가 주어졌을 때 인접하지 않는 두 vertex를 연결하여 clique를 탐색하는 line addition 방법, 그리고 인접하는 두 vertex를 연결하는 edge가 존재하는 clique가 주어졌을 때 그 edge를 제거한 그래프 G의 clique를 탐색하는 line removal 방법이 있다. 이 방법에는 Robert의 알고리즘이 해당한다. Backtracking technique는 탐색 clique에 이르지 못하는 branch를 사전에 배제하여 탐색과정을 최적화하는 branch-and-bound방법을 사용한다. 여기에는 Bron의 알고리즘이 해당한다. 마지막으로 stack operation 방법인데 그래프를 표현하는 수식을 스택을 이용하여 조작함으로써 크기가 큰 문제에 대해서도 효과적으로 대처할 수 있는 알고리즘이다. 이 방법에는 Akkoynulu 알고리즘이 해당한다.

Bierstone의 알고리즘은 adjacency matrix M이 0인 경우에 정상적으로 동작하지 못하며 멈추지 않는다. 그리고 maximal이 아닌 clique가 생성되거나 바로 이전에 생성된 subgraph의 부집합인 clique를 삭제시키지만 그대로 무시한다. 또한 고립된 point들에 대해서는 clique를 생성하지 못한다. Bierstone의 알고리즘은 비교적 큰 clique들이 존재하는 그래프의 적용에 알맞다. Gordon의 알고리즘은 Bierstone 알고리즘의 2단계에 그래프의 vertex 수를 조사하는 과정을 삽입함으로써 알고리즘이 끝나지 않는 위험을 배제하였고, 12단계에 clique가 maximal일 경우에 이미 생성된 clique중에서 부분집합이 되는 clique를 제거하는 과정을 추가함으로써 maximal이 아닌 clique가 생성되는 오류를 처리하였다. Gordon 알고리즘은 비교적 큰 clique를 갖는 그래프에 잘 적용된다. 그러나 Gordon의 알고리즘은 고립된 point들은 clique로 생성하지 못한다. Born의 알고리즘은 알고리즘이 생성하는 clique의 비율이 그래프의 크기에 관계가 없으며, 3*K의 regular 그래프에서 Bierstone의 알고리즘은 생성되는 set들의 중복이 없으며 이전에 생성된 set들을 참조하거나 유지할 필요가 없다. 이것은 maximal subgraph set이 겹치지 않도록 작은 문제로 체계적으로 줄이기 때문이다. 또한 크기가 큰 문제에 대해서도 기억공간의 크기에 큰 영향을 받지 않는다. Akkoynulu 알고리즘은 그래프의 clique가 주어졌을 때 그래프의 선을 제거하거나 추가하여 clique를 구하는 방법이다. 이런 알고리즘은 같은 점을 갖는 모든 그래프의 순서를 생성하는 문제, 그래프의 clique들을 요구하는 cluster의 분석 문제와 같은 그래프 이론적 cluster 방법에서 발생하는 특수한 문제에 적용될 수 있다.

참 고 문 헌

[1] J.A.Bondy and U.S. R. Murty, Graph Theory with Ap-

plications, American Elsevier Publishing Co., Inc., 1976.

[2] Bierstone, E., "Cliques and generalined cliques in a finite linear graph," JACM, Vol.16, No.4, Oct. 1969, pp. 571－588.

[3] Auguston, J. G. and Minker, J., "An analysis of some theoretical cluster Techniques," journal of the Association for Computing Machinary, Vol. 17, No.4, pp. 571－588, Oct. 1970.

[4] E. A. Akkoyunlu, "The enumeration of maximal cliques of large graphs," SLAM J. Comput., Vol. 2, No. 1, pp. 1-6, Mar. 1973.

[5] Gordon D. Mulling and D. G. Corneil, "Corrections to Bierston's algotithm for generating clique,", J. ACM, Vol. 19, No. 2, pp. 224-247, Apr. 1972.

[6] Coen Born and Joep Kerbosch, "Finding all cliques of an undirected graph : Algorithm 457," Comm. ACM, Vol. 16, No. 9, pp. 575-577, Sep. 1973.

[7] R. E. Osteen, "Clique detection algorithms based on line addition and line removal," SLAM J. Appl. Math., Vol. 26, No. 1, pp. 126-135, Jan. 1974.

[8] R. E. Tarjan and A. E. Troanowski, "Finding a maximum independent sets," SLAM J. Comput., Vol. 6, No. 3, pp.537－546, Sep. 1977.

[9] S. Tsukiyama, M. Ide, H. Ariyoshi and I. Sirakawa, "A new algorithm for generating all the maxiaml independent sets." SLAM J. Vol. 6, No. 3, pp. 505-517, Sep. 1977.